Re 09/970,409 – EAST 1.2

Type	L #	Hits	Search Text	DBs	Time Stamp
IS&R	L3	8	(("6026233") or ("6311323") or ("6305008") or ("6266665")).PN.	USPAT; EPO; JPO; DERWENT; IBM TDB	2002/03/21 09:19
BRS	L2	9	5485618.URPN.	USPAT; EPO; JPO; DERWENT; IBM TDB	2002/03/21 09:20
BRS	L4	52	("5006992" | "5263174" | "5377318" | "5485618" | "5537630" | "5628016" | "5680630" | "5734749" | "5740444" | "5790778" | "5798757" | "5813019" | "5844554" | "5845300" | "5911075" | "5924089" | "5959629" | "6026233").PN.	USPAT; EPO; JPO; DERWENT; IBM TDB	2002/03/21 09:54
BRS	L5	59	("4459678" | "4622013" | "4648062" | "4789962" | "4806919" | "4890257" | "4899276" | "4964077" | "4970678" | "4972328" | "4992972" | "5029113" | "5040131" | "5123087" | "5155806" | "5157768" | "5179654" | "5255363" | "5303146" | "5317686" | "5317688").PN.	USPAT; EPO; JPO; DERWENT; IBM TDB	2002/03/21 09:21
BRS	L6	9	("4989145" | "5493678" | "5513305").PN.	USPAT; EPO; JPO; DERWENT; IBM TDB	2002/03/21 09:26
BRS	L7	3064	edit$3 same tool	USPAT; EPO; JPO; DERWENT; IBM TDB	2002/03/21 09:55
BRS	L8	3253	auto or automatic$4)same(generat$3 or creat$3)same(window or dialog adj1 box	USPAT; EPO; JPO; DERWENT; IBM TDB	2002/03/21 09:58
BRS	L9	193	l7 and l8	USPAT; EPO; JPO; DERWENT; IBM TDB	2002/03/21 10:01
BRS	L10	1029	(assist$3 or select$3 or sugest$3) same l8	USPAT; EPO; JPO; DERWENT; IBM TDB	2002/03/21 10:00
BRS	L11	104	l7 and l10	USPAT; EPO; JPO; DERWENT; IBM TDB	2002/03/21 10:01
BRS	L12	31	compil$5 and l11	USPAT; EPO; JPO; DERWENT; IBM TDB	2002/03/21 10:53
IS&R	L13	149	("717/110-113").CCLS.	USPAT; EPO; JPO; DERWENT; IBM TDB	2002/03/21 10:53
IS&R	L14	1159	("345/710,714-716,762-764,781,808,809,825").CCLS.	USPAT; EPO; JPO; DERWENT; IBM TDB	2002/03/21 10:56
IS&R	L15	1258	("707/512,530,531,534").CCLS.	USPAT; EPO; JPO; DERWENT; IBM TDB	2002/03/21 10:59
BRS	L16	31	l11 and (l13 or l14 or l15)	USPAT; EPO; JPO; DERWENT; IBM TDB	2002/03/21 10:59
BRS	L17	82	l10 and (l13 or l14 or l15)	USPAT; EPO; JPO; DERWENT; IBM TDB	2002/03/21 10:59

DOCUMENT-IDENTIFIER: US 6314559 B1
TITLE: Development system with methods for assisting a user with inputting source code

ABPL:
A visual development system having an interface which assists a user with input of source code expressions and statements during creation of a computer program is described. The interface includes an Integrated Development Environment (IDE) interface having a code editor with "Code Completion" and "Code Parameter" features for displaying context sensitive pop-up windows within a source code file. Code Completion is implemented at the user interface level by displaying a Code Completion dialog box after the user enters a record or class name followed by a period. For a class, the dialog lists the properties, methods and events appropriate to the class. For a record or structure, the dialog lists the data members of the record. To complete entry of the expression, the user need only select an item from the dialog list, whereupon the system automatically enters the selected item in the code. Code completion also operates during input of assignment statements. When the user enters an assignment statement for a variable and presses a hot key (e.g., <ctrl><space_bar>), a list of arguments valid for the variable is displayed. Here, the user can simply select an argument to be entered in the code. Similarly, the user can bring up a list of arguments when typing a procedure, function, or method call and needs to add an argument. In this manner, the user can view the required arguments for a method as he or she enters a method, function, or procedure call.

BSPR:
A program called a "compiler " translates these instructions into the requisite machine language. In the context of this translation, the program written in the high-level language is called the "source code" or source program. The ultimate output of the compiler is an intermediate module or "object module," which includes instructions for execution by a target processor. In the context of Borland's Turbo Pascal and Object Pascal, the intermediate module is a Pascal "unit" (e.g., .TPU file). Although an object module includes code for instructing the operation of a computer, the object module itself is not usually in a form which may be directly executed by a computer. Instead, it must undergo a "linking" operation before the final executable program is created.

BSPR:
Linking may be thought of as the general process of combining or linking together one or more compiled object modules or units to create an executable program. This task usually falls to a program called a "linker." In typical operation, a linker receives, either from the user or from an integrated compiler, a list of modules desired to be included in the link operation. The linker scans the object modules from the object and library files specified. After resolving interconnecting references as needed, the linker constructs an executable image by organizing the object code from the modules of the program in a format understood by the operating system program loader. The end result of linking is executable code (typically an .EXE file) which, after testing and quality assurance, is passed to the user with appropriate installation and usage instructions.

DOCUMENT-IDENTIFIER: US 6311323 B1
TITLE: Computer programming language statement building and information tool

ABPL:
An intelligent real time tool to assist a computer programmer during the writing and/or maintenance of a computer program. The tool generates assist windows that contain program related information that the programmer can use to construct a programming language statement and/or to obtain real time information about a programming language statement. An assist window can be automatically displayed as determined by the tool itself, and/or manually displayed on demand in response to a user input command. An assist window displays two general categories of information including but not limited to selection menu information based on a partial compilation of all programming language statements, and informational displays based on a partial compilation and a reverse parse of an immediate programming language statement. The statement generating tool assist windows are non-intrusive to programmer

input and can be ignored by the programmer by merely continuing to type an immediate programming language statement without interacting with the assist windows that are proximate the programming language statement being constructed by the programmer.

BSPR:
As a computer program evolves during the coding process, accuracy becomes a particular problem due in part to the arcane syntax of programming languages in characteristic of programmer-declared object entities in a computer program, such as variable names, parameter names, structure names, structure members, object names, object property types, and the like. Not only must references to a declared entity be spelled correctly, the declared entity must also be used in the correct context and syntax if the resulting computer program is expected to compile without errors and/or execute as intended.

DOCUMENT-IDENTIFIER: US 6181336 B1
TITLE: Database-independent, scalable, object-oriented architecture and API for managing digital multimedia assets

DEPR:
For example, suppose a developer of a complex video editing tool would like such tool to be integrated with the asset management facilities provided by the present invention without making any changes therein. Alternatively, suppose a tool developer other than the developer of the complex video editing tool would like to provide a utility program that allows the complex video editing tool to use the asset management facilities provided by the present invention. In either case, instead of modifying the code and rebuilding the complex video editing tool so that it is directly integrated with an embodiment of the present invention, a relatively small stand-alone AMS aware checkin/checkout tool can be developed. This stand-alone tool acts as an interface between the asset management system of the present invention and the multimedia file management system of the complex video editing tool . In this fashion, an existing unmodified content creation tool can be seemingly integrated into the production system environment of the present invention. Examples of this concept will be subsequently provided herein with reference to FIG. 2.

DEPR:
Some of the attribute values, such as time, date, and object size is typically filled-in automatically under program control. Other attributes are typically supplied by the creator or user of the asset management tool 102. Thus, support code for user interaction, typically implemented in the form of a dialog box or the like, is provided in step 1008. Additionally, a selection of common user interface functions may be provided by the program resources 100 of the present invention to facilitate the implementation of such services and to provide a common look and feel among diverse asset management tools 102. Note that the programming code of step 1008 consults the Data Model Manager for information about the asset type inserted in step 1006 to determine what attributes to request from the user. The programming code also uses this asset type (or data model) information in the Data Model Manager to ensure that values supplied by the user fall within the allowed ranges.

DOCUMENT-IDENTIFIER: US 6069627 A
TITLE: Extender user interface

DEPR:
FIG. 4 is a block diagram that illustrates the development environment of the present invention. A development computer 400 executes a Rapid Application Development (RAD) tool comprised of a number of different computer programs or modules, including a graphical user interface (GUI) 402, project manager 404 and associated builder 406, form editor 408 for constructing HTML forms 410, code editor 412 for constructing scripts 414, debugger 416, SQL painter 418 for constructing queries 420, RDBMS extender guide 422, and RDBMS extender user interface 424, as well as a browser 426, web

server 428, VAB-II runtime module 430, and RDBMS 432. The RAD tool displays a user interface on a monitor 434 attached to the development computer 400, which includes, inter alia, a project window 436, form editor window 438, control pad 440, code editor window 442, debugging window 444, extender user interface window 446, extender guide window 448, SQL painter window 450, as well as a browser window 452.

DEPR:
FIG. 7G illustrates a smart guide window 448 for generating extender project files. This smart guide window 448 displays the extender object name 760, a build path 762, an attributes list box 764 containing attributes, and a methods list box 766 containing methods. If the information provided by the smart guide is correct, a user selects the finish button 768, and then the extender smart guide 422 automatically generates the extender project files.

DEPR:
Block 806 represents the extender smart guide 422 compiling the source code to interface to a generic extender.

CLPV:
compiling the source code to interface to an extender

DOCUMENT-IDENTIFIER: US 6026233 A
TITLE: Method and apparatus for presenting and selecting options to modify a programming language statement

ABPL:
An intelligent real time tool to assist a computer programmer during the writing and/or maintenance of a computer program. The tool generates assist windows that contain program related information that the programmer can use to construct a programming language statement and/or to obtain real time information about a programming language statement. An assist window can be automatically displayed as determined by the tool itself, and/or manually displayed on demand in response to a user input command. An assist window displays two general categories of information including but not limited to selection menu information based on a partial compilation of all programming language statements, and informational displays based on a partial compilation and a reverse parse of an immediate programming language statement. The statement generating tool assist windows are non-intrusive to programmer input and can be ignored by the programmer by merely continuing to type an immediate programming language statement without interacting with the assist windows that are proximate the programming language statement being constructed by the programmer.

BSPR:
The above identified problems are solved and an advancement made in the programming tool field by the computer programming language statement building and information tool of the present invention. The present invention generates assist windows that contain program related information for use by a programmer to construct a programming language statement and/or to obtain real time information about a programming language statement. Constructing a programming language statement is a process referred to as statement building. The assist windows can be automatically displayed as determined by the tool itself. Alternatively, and/or in combination with the automatic features, a programmer can manually request that an assist window be displayed on demand. The automatic display feature can be enabled and disabled independent of and without inhibiting the manually requested assist window display feature of the present invention.

BSPR:
An assist window displays two general categories of information including, but not limited to, selection menu information, and informational display. One important feature of the statement generating tool is that assist windows are non-intrusive and can be ignored by the programmer by merely continuing to type

an immediate programming language statement at a present character position cursor location without interacting with the assist windows. The assist windows continue to appear, disappear, and/or update in a location proximate to but out of the way of the present character position cursor location so long as the automatic assist window display feature is enabled.

DOCUMENT-IDENTIFIER: US 6023715 A
TITLE: Method and apparatus for creating and organizing a document from a plurality of local or external documents represented as objects in a hierarchical tree

BSPR:
The above and other objects are achieved with an inventive authoring tool which allows for creation of documents having relatively limitless nesting of division within the main document. Each division within the main document may have its own style, format and content. Division content may even be located externally of the main document and the author's computer system. Graphic elements, in the form of divider tabs, are associated with each division in the main document structure. The tabs can be displayed to provide a visual reference as to location within the main document, and, may be manipulated graphically to both navigate through the document and reorganize the document. In addition, the main document may be viewed in a contiguously scrolling manner from a single view. The implementation of divisions within a master document enables concurrent creation and/or editing of divisions within the document. For implementations including a vertical scroll bar, a display window adjacent the scroll bar dynamically displays the name of the division and a page number at the current cursor position within the scroll bar.

DEPR:
Alternatively, the user may select the divider tab 505 of an existing division and, by subsequently clicking the right mouse button, access the division menu 506, as illustrated in FIG. 5D. Selection of the "Division Properties" option from menu 506 will cause the dialog box 508 of FIG. 5E to appear. As illustrated, dialog box 508 includes additional dialog boxes which allow the user to select a name, page style, and a starting location for the division, as well as to select visibility and color options for the accompanying divider tab. Selection of the "Quick Division" option from menu 506 automatically creates a new division having all the characteristics and attributes of the division whose divider tab was selected . Selection of the "Combine Divisions" option of menu 506 causes a dialog box to appear which enables the user to select and combine any of the divisions existing within the current document, from a displayed list thereof. Selection of the "Delete Division" option automatically deletes the division whose divider tab was selected . Selection of the "Group Tabs" option for menu 506 causes a nesting effect, i.e., creation of a pseudo-parent division level between the selected division and its immediately proceeding parent division. Selecting the "Collapse Nested Group" division causes the divider tabs currently displayed within a group tab to become hidden.

DOCUMENT-IDENTIFIER: US 5850561 A
TITLE: Glossary construction tool

BSPR:
A number of semi-automated glossary construction tools have been developed or proposed for facilitating the translation process using parallel texts. For example, a number of researchers have proposed finding candidate translations from parallel texts aligned at the sentence level by allowing a user to search for a desired term in the aligned parallel texts. For a discussion of proposed techniques for generating translation glossaries using parallel texts aligned at the sentence level, see, for example, William Ogden and Margarita Gonzales, "Norm--A System for Translators," Presentation, ARPA Workshop on Human Language Technology, p. 223 (Mar. 21-24, 1993); and Frank Smadja, "How to Compile a Bilingual Collocational Lexicon Automatically," AAAI Workshop on Statistically-Based Natural Language Processing Techniques, (July, 1992).

DEPR:
The glossary construction tool 10 includes a document data base 150 for storing one or more source text documents 152, which are the documents to be translated into the target language using the glossary construction tool 10. As previously indicated, source text documents 152 may be received for translation from a customer in the form of a paper document 50 or as a computer-readable electronic file, for example, from a source computer 60. It is noted that source text documents 152 received by the glossary construction tool 10 in an electronic format may have to be processed by a mark-up language filter 190, which will process files produced using standard text editing programs, in a known manner, to facilitate further processing.

DEPR:
In order to facilitate scanning of the concordance lines presented in the concordance window 630 for missing candidate terms, all occurrences of the selected candidate term in the concordance lines are preferably sorted such that identical preceding contexts of the selected term are grouped together. For example, for the candidate term "point", all occurrences of the phrase "starting point" are preferably grouped together. In this manner, the noun phrase "starting point" may be identified by the user as a technical term for addition to the final terminology list in the window 610, even though the phrase "starting point" failed to appear in the automatically generated candidate terminology list because the word "starting" is tagged by the part-of-speech tagger 164 as a verb. Preferably, the user can identify a technical term to be added from the concordance region 630 to the final terminology list in the window 610 by graphically "blocking" the desired term, in a known manner, and depressing the add button 670, or alternatively, by entering the desired term using a keyboard command.

DEPR:
In this manner, a user can select a term to be translated from the glossary window 910 and review and evaluate the candidate translations appearing in window 920 which are automatically generated by the glossary development tool 170. Thus, the user can select the appropriate translation to be entered in the translation glossary 156 for the selected source term or find translations that are missing from the candidate translation list. In order to facilitate the user's evaluation, the bilingual concordances for a selected candidate translation 940, such as the candidate translation "menu caracteres" selected in FIG. 9, are presented in concordance window 930. As previously indicated, the glossary development tool 170 will access the entry in the bilingual concordance list 184 for the selected candidate translation and present all occurrences of the selected source term in context in the source text document of the aligned text pair, and show the corresponding position of the translation in the target text document of the aligned text pair 154.

DEPR:
In particular, although the above embodiment has been described in a translation environment, the terminology list development tool 160 can be utilized to construct terminology lists in other applications as well, such as technical writing, book indexing, hypertext linking, natural language interfaces, text categorization and indexing in digital libraries and information retrieval. In addition, the glossary development tool 170 can be useful for information retrieval in multilingual text collections, and for verification of translation consistency at the proofreading or editing step of a translation job, after the document has been translated. For example, if the glossary development tool 170 identified the translation of a particular term in the same way for four out of five occurrences, a violation of the consistency requirement can be identified by a unique translation in the fifth occurrence.

CLPR:
1. A system for compiling a terminology list from a source document, said system comprising

DOCUMENT-IDENTIFIER: US 5598524 A
TITLE: Method and apparatus for improved manipulation of data between an application program and the files system on a computer-controlled display system

BSPR:
Upon viewing FIGS. 1a-1d and 2a-2d, it is apparent that there is a dichotomy between the two techniques. Users become used to the manipulation of files in the manner which is illustrated with reference to FIGS. 1a-1d, however, the user must learn the use of a second tool known as the "Edit " pull-down menu illustrated as 230 in FIGS. 2a-2d in order to perform manipulation of information between windows and/or application programs and/or files. Thus, there is a need for improved manipulations of various types of dam, especially between application programs than techniques which exist in the prior art.

BSPR:
These and other objects of the present invention are provided for by a method and apparatus for a user selecting an item in a first window wherein the window is under control of a first process. The user may drag a selected item to a second window, wherein the second window is under control of a file system manager of the computer system. The user will deselect the first item, and the file system manager will automatically create a file containing the item which was selected in the first window . Various types of data may be transferred in this way, such as text, graphic data, sound dam, video, or other data items in the computer system. A representation of the item is presented on the display of the computer system to indicate the type of item which the file represents. The representation may include an item and an identifying file name which is specified by the first application program.

DEPR:
Note that the following discussion of the methods and apparatus of the preferred embodiment discussed herein will refer specifically to a series of routines which are compiled, linked, and then run as object code in computer system 300 during run-time. It can be appreciated by one skilled in the art, however, that the foregoing methods and apparatus may be implemented in special purpose hardware devices, such as discrete logic devices, large scale integrated circuits (LSI's), application-specific integrated circuits (ASIC's), or other specialized hardware. The description here has equal application to other apparatus having similar function.

DOCUMENT-IDENTIFIER: US 5579469 A
TITLE: Global user interface

BSPR:
Help's roots lie in Wirth's and Gutknecht's Oberon system, described in N. Wirth and J. Gutknecht, "The Oberon System", Software Practice and Experience, September 1989, vol 19, no. 9, pp 857-894 and in Martin Reiser, The Oberon System, Addison Wesley, New York 1991. Oberon is an attempt to extract the salient features of Xerox's Cedar environment, described in W. Teitelman, "A Tour through Cedar", IEEE Software 1, no. 2, pp. 44-73, and implement them in a system of manageable size. It is based on a module language, also called Oberon, and integrates an operating system, editor, window system, and compiler into a uniform environment. Its user interface is especially simple: by using the mouse to point at text on the display, one indicates what subroutine in the system to execute next. In a normal UNIX shell, one types the name of a file to execute; instead in Oberon one selects with a particular button of the mouse a module and subroutine within that module, such as Edit.Open to open a file for editing. Almost the entire interface follows from this simple idea.

DEPR:
A typical shell window in a traditional window system permits text to be copied from the typescript and presented as input to the shell to achieve some son of history function: the ability to re-execute a previous command. Help instead tries to predict the future: to get to the screen commands and text that will be useful later. Every piece of text on the screen is a potential command or argument for a command. Many of the basic commands pull text to the screen from the file system with a minimum of fuss. For example, if Open is executed without an argument, it uses the file name containing the most recent selection (the rule of defaults). Thus one may just point with the left button at a file name and then with

the middle button at Open to edit a new file. Using all four of the rules above, if Open is applied to a null selection in a file name that does not begin with a slash /), the directory name is extracted from the file name in the tag of the window and prepended to the selected file name. An elegant use of this is in the handling of directories. When a directory is Opened, help puts the its name 121, including a final slash, in the tag and just lists the contents 123 (i.e., the names of the files in the directory) in the body. For example, as shown in FIG. 3, by pointing at dat.h 301 in the source file /usr/rob/src/help/help.c and executing Open, a new window 305 is created containing the contents of /usr/rob/src/help/dat.h: all it takes is two button clicks. Making any non-null selection disables all such automatic actions: the resulting text is then exactly what is selected .

DEPR:
It is possible to execute any external operating system command. If a command is not a built-in like Open, it is assumed to be an executable file and the arguments are passed to the command to be executed. For example, if one selects with the middle button the text grep ` main` /sys/src/cmd/help/*.c the traditional grep command will be executed. Again, some default rules come into play. If the tag line 105 of the window 103 containing the command has a file name and the command does not begin with a slash, the directory of the file will be prepended to the command. If that command cannot be found, the command will be searched for in the conventional directory/bin. The standard input of the commands is connected to /dev/null; the standard and error outputs are directed to a special window, called Errors, that will be created automatically if needed. The Errors window is also the destination of any messages printed by the built-in commands.

DEPR:
When help starts it loads a set of `tools` into the right hand column of its initially two-column screen. These are files with names like /help/edit/stf (the stuff that the help editor provides), /help/mail/stf, and so on. Each is a plain text file that lists the names of the commands available as parts of the tool, collected in the appropriate directory. A help window on such a file behaves much like a menu, but is really just a window on a plain file. The useful properties stem from the interpretation of the file applied by the rules of help; they are not inherent in the file. Turning to FIG. 4, mail is read by executing the program headers in the mail tool, that is, I click the middle mouse button on the word headers 403 in window 401 containing the file/help/mail/stf. This executes the program /help/mail/headers by prefixing the directory name of the file /help/mail/stf, 405, collected from tag 407, to the executed word, headers. This simple mechanism makes it easy to manage a collection of programs in a directory.

DEPR:
To create a new window, a process just opens /mnt/help/new/ctl, which places the new window automatically on the screen near the current selected text, and may then read from that file the name of the window created, e.g. /mnt/help/8. The position and size of the new window is, as usual, chosen by help.

DEPR:
A couple of observations about this example. First, help provided all the user interface. To turn a compiler into a browser involved spending a few hours stripping the code generator from the compiler and then writing a half dozen brief shell scripts to connect it up to the user interface for different browsing functions. Given another language, we would need only to modify the compiler to achieve the same result. We would not need to write any user interface software. Second, the resulting application is not a monolith. It is instead a small suite of tiny shell scripts that may be tuned or toyed with for other purposes or experiments.

DOCUMENT-IDENTIFIER: US 5877758 A
TITLE: System and method for using a slider control for controlling parameters of a display item

DEPR:

Those skilled in the art will recognize conventional display items or elements shown in the screen displays for the Workload Planner view. As best shown in FIG. 2A, menu bar 102, which is located near the top of the screen, contains several pull-down menus 104. The pull-down menus on the menu bar 102 allow the user to perform such tasks as editing, formatting and viewing files. A main tool bar 106, which is located below the menu bar 102, allows the user to perform common commands by selecting buttons located on the tool bar. A status bar 108 is located near the bottom of the screen and contains a return to a Program Manager button 110, and buttons for switching to other open applications 112.

DEPR:
Typical usage of a slider control is as follows: The control is created . If the control is specified in a dialog box template, creation is automatic when the dialog box is created . Alternatively, the Create member function can be used to create the control as a child window of any window . Various Set member functions can be called to set values for the control. The changes that can be made include setting the minimum and maximum positions for the slider, drawing tick marks, setting a selection range, and repositioning the slider.

DOCUMENT-IDENTIFIER: US 5608898 A
TITLE: Development system with methods for maintaining data integrity of information stored as a data record in a database with a copy of the information displayed on a screen device

BSPR:
In accordance with the invention, if during the course of creating a form, an expression assigned to a branch or conclusion references a form field which does not exist, the system automatically creates a new field which adopts the established name. Subsequently, a field may be placed on the form to hold that name; however, if no field is assigned on the form, the system automatically prompts for a value at the appropriate place during the completion of the form. The prompt for such a field presents a prompt window that requests selection of a value for the question that does not appear on the form; however, a value is required for that field since continued prompting in the form is dependent on the value selected .

DEPR:
The Tools menu includes Form, Tree, Stack and Link commands for invoking the respective tools. Form selects the Form tool and loads the Form Tool Operations as the main menu, as shown in FIG. 6. Tree selects the Tree tool and loads the Tree Tool Operations as the main menu as shown in FIG. 7. Stack selects the Stack tool and loads the Stack Tool Operations as the main menu as shown in FIG. 8. Link selects the Link tool, which employs dialogue windows to create and/or edit links.

DEPR:
As shown in FIG. 6, the Form Tool window includes operations of Form, Edit, Objects, Properties, Alignment, Font, Borders, Fill Pattern, Line Width, Protection, Field, Name/Text, Help, View, and Tools. The submenu commands of each will be described next.

DEPR:
The form tool provides the following capabilities: (a) creation of a new form; (b) adding new objects to a form; (c) renaming, sizing and scrolling forms; (d) finding forms that contain a specified field; (e) selecting, moving and sizing form objects; (f) editing form objects with the clipboard; (g) changing the field referenced by a field object; (h) changing the names of field and text objects; (i) adding help text to be displayed for a field object; (j) changing the display format of a field object; (k) changing the alignment of text within field objects and text objects; (l) changing the character fonts of text objects and field objects; (m) controlling which, if any, borders are drawn around objects; (n) controlling whether the field name is displayed in a field object; and (o) protecting field objects both from override by the operator or display of the tree associated with the field object.

DEPR:
FIG. 21 shows a dialog box that allows for the automatic generation of the values for fields. This dialog box appears whenever the operator changes the type of a field to either "selection list" or "check box"

using the "Field Type" command on the "Properties" menu shown in FIG. 6. The automatic determination of the values looks at values that can be attached from the tree, values that are used in a tree which employs the field for determination of the other tree's value, or finally automatic creation of the values by looking at the values that can be brought from the records of a database. If automatic is not selected, then the new values are manually entered in the edit box under "New Value" and then added to the list in the box called "Values".

DEPR:
In a preferred embodiment, the object bar includes a field object 553, a button object 555, a table object 557, a text object 559, a rectangle object 561, a rounded rectangle object 563, a line object 565, and a graphics (bitmap) object 567. The object bar 550 also includes a stack tool 569 for assigning event trees (described below) to a stack, and a link tool 571 for linking or connecting objects to external databases. Finally, the object bar 550 includes a close tool 551 for exiting from the form edit mode.

DEPR:
The "customers" link is a simple link connected to fields. To connect application fields to CUSTOMRS.DB (a Paradox table), the user performs the following steps. The Customers form 700 of FIG. 38A is made the active form (if not already). The link buttons (Enter, Next, Previous, Delete, Clear, Top, and Bottom) are automatically placed on the active form after the link is created . Next, the user selects Link tool 571 from the object bar. The Data Links dialog box 900 appears as shown in FIG. 41A. After the external target (in this example, Paradox) is specified (e.g., in the Link Creation list), then invokes the Create command. A Link Creation dialog box 910 specific for the target (Paradox) appears as shown in FIG. 41B. The user enters customers in the Link Name text box 911. Any link name that has not already been used may be entered; however, it is helpful to choose a name that helps one remember the purpose of the link (e.g., "cust" for customer).

DEPR:
To specify the circumstances when the link will update the data file, the user selects the Options button of the target (e.g., Paradox) Link Creation dialog box . The Optional Link Capabilities dialog box 940 appears as shown in FIG. 41F. Auto Update and Auto Insert define when changes are written to the database.

DEPR:
Continuing with the link of the present example, the user checks Restricted Range, and leaves Auto Locate checked. The Locate Indexes dialog box, the Optional Link Capabilities dialog box, and the Paradox Link Creation dialog box are closed. A Link Automatic Buttons dialog box 960 appears, as shown in FIG. 41H. By selecting the buttons that the user wants and then clicking OK, the system creates link navigation buttons complete with their event trees. Each choice in the list has a corresponding @function. For instance, Top, Bottom, Previous, Next, Clear, Delete, and Store buttons are useful for browsing and entering data.

DEPL:
block selection: In the Form Tool, lets the user select multiple objects in order to perform editing operations, assign or revise properties, or reposition the selected fields as a group of objects.

DOCUMENT-IDENTIFIER: US 6246404 B1
TITLE: Automatically generating code for integrating context-sensitive help functions into a computer software application

DEPR:
FIG. 2 illustrates a method 34 for creating help functions in an illustrative embodiment of the present invention. Method 34 is implemented in a dialog box help editor which is executed on computer system 10. A computer software application for which context-sensitive help functions are being created need

not be complete or executable to use method 34. A set of dialog boxes contained within a computer software application are located at step 36. In response to a first selection input, a dialog box from the set of located dialog boxes is displayed on a display device at step 38. The displayed dialog box typically includes one or more graphical control objects. In response to a second selection input, a help function template for a graphical control object selected in the displayed dialog box is created at step 40. The selection inputs at steps 38-40 are typically made in response to inputs made by a user with a mouse or other pointing device. In an alternative embodiment of the present invention, the selection inputs are made by another computer software application executing on the local computer system 10, or a remote computer system (e.g., with a message or a function call). Help information data is received and incorporated into the help function template for the selected graphical control object at step 42. In an illustrative embodiment of the present invention, the help information data is input by a help writer. In an alternative embodiment of the present invention, the help information data is read from a file, or sent by another computer software application. Computer source code for a context-sensitive help function for the selected graphical control object is generated at step 44. The source code is generated automatically and includes the help information data received for the help function template. Each of the steps of method 34 are explained below.

DEPR:
In response to a selection input of a graphical control at step 40, the dialog-box -help-editor 46 creates a new help topic template using a help-topic-creation -module 54. The help-topic-creation -module 54 automatically assigns a Help ID (e.g., IDH_X_Y) for the help topic template to a selected control by using a combination of the dialog box ID (X) and the control ID (Y) to ensure uniqueness. For example, if the dialog box ID is 100 and the control ID is 102, the Help ID for the selected control would become a combination of the two (e.g., IDH_100_102). However, other methods may also be used for assigning Help IDs. A user may choose to assign his or her own ID to a control by using the custom Help ID option available in the help-topic-creation -module 54. For example, a user may create a Help ID for a graphical button as IDH_BUTTON.